URL Access Manager Reference

(Not Recommended)

Carbon > Networking

2006-07-13

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

URL Access Manager Reference (Not Recommended)

Framework: Carbon/Carbon.h

Declared in URLAccess.h

Overview

Important: URL Access Manager is deprecated as of Mac OS X v10.4. You should use CFNetwork instead (as described in *CFNetwork Programming Guide*).

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

The URL Access Manager is an API that you can use to perform data transfer to and from a URL from within your application. It includes support for automatic decompression of compressed files and for automatic file extraction from Stuffit archives (with version 5.0 of Stuffit).

The URL Access Manager allows you to use any of the following protocols during download operations: File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), secure Hypertext Transfer Protocol (HTTPS), or a URL representing a local file (begins with `file://`). You might use the latter to test your application on a computer that does not have access to an HTTP or FTP server. For upload operations, you must use an FTP URL.

This document describes the URL Access Manager API through version 2.0.3.

Functions by Task

Getting Information About the URL Access Manager

`URLGetURLAccessVersion` (page 44) Deprecated in Mac OS X v10.4

Determines the version of URL Access Manager installed on the user's system. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

Creating and Disposing of a URL Reference

`URLDisposeReference` (page 36) Deprecated in Mac OS X v10.4

Disposes of the memory associated with a URL reference. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLNewReference` (page 45) Deprecated in Mac OS X v10.4
Creates a URL reference. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

Getting and Setting Information About a URL

`URLGetProperty` (page 42) Deprecated in Mac OS X v10.4
Obtains the value of a URL property. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLGetPropertySize` (page 43) Deprecated in Mac OS X v10.4
Determines the size of a URL property. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLSetProperty` (page 48) Deprecated in Mac OS X v10.4
Sets the value of a URL property. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

Performing Simple Data Transfer

`URLDownload` (page 37) Deprecated in Mac OS X v10.4
Downloads data from a URL specified by a URL reference. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLSimpleDownload` (page 49) Deprecated in Mac OS X v10.4
Downloads data from a URL specified by a character string. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLSimpleUpload` (page 51) Deprecated in Mac OS X v10.4
Uploads a file or directory to an FTP URL specified by a character string. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLUpload` (page 53) Deprecated in Mac OS X v10.4
Uploads a file or directory to an FTP URL specified by a URL reference. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

Getting More Control Over Data Transfer Operations

`URLGetBuffer` (page 38) Deprecated in Mac OS X v10.4
Obtains the next buffer of data in a download operation. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLGetDataAvailable` (page 40) Deprecated in Mac OS X v10.4
Determines the amount of data currently available for retrieval in a download operation. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLOpen` (page 46) Deprecated in Mac OS X v10.4
Opens a URL and starts an asynchronous download or upload operation. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLReleaseBuffer` (page 48) Deprecated in Mac OS X v10.4
Releases a buffer. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

Terminating Data Transfer Operations

`URLAbort` (page 35) Deprecated in Mac OS X v10.4
: Terminates a data transfer operation. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

Getting Data Transfer Information

`URLGetCurrentState` (page 39) Deprecated in Mac OS X v10.4
: Determines the status of a data transfer operation.

`URLGetError` (page 41) Deprecated in Mac OS X v10.4
: Determines the error code of a failed data transfer operation. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLGetFileInfo` (page 42) Deprecated in Mac OS X v10.4
: Obtains the file type and creator of a file. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`URLIdle` (page 45) Deprecated in Mac OS X v10.4
: Gives the URL Access Manager time to refill its buffers during download operations. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

Working With URL Access Manager Callbacks

`DisposeURLNotifyUPP` (page 31) Deprecated in Mac OS X v10.4
: Disposes of a UPP to your data transfer event notification callback. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`DisposeURLSystemEventUPP` (page 31) Deprecated in Mac OS X v10.4
: Disposes of a UPP to your system event notification callback. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`InvokeURLNotifyUPP` (page 32) Deprecated in Mac OS X v10.4
: Invokes your data transfer event notification callback. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`InvokeURLSystemEventUPP` (page 33) Deprecated in Mac OS X v10.4
: Invokes your system event notification callback. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`NewURLNotifyUPP` (page 34) Deprecated in Mac OS X v10.4
: Creates a UPP to your data transfer event notification callback. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

`NewURLSystemEventUPP` (page 35) Deprecated in Mac OS X v10.4
: Creates a UPP to your system event notification callback. (Deprecated. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

Callbacks

URLNotifyProcPtr

Defines a pointer to your notification callback function that handles certain data transfer events that occur during data transfer operations.

```
typedef OSStatus (*URLNotifyProcPtr)
(
    void * userContext,
    URLEvent event,
    URLCallbackInfo * callbackInfo
);
```

If you name your function `MyURLNotifyProc`, you would declare it like this:

```
OSStatus MyURLNotifyProc (
    void * userContext,
    URLEvent event,
    URLCallbackInfo * callbackInfo
);
```

Parameters

userContext

A pointer to application-defined storage that your application previously passed to the function `URLOpen` (page 46). Your application can use this to set up its context when your notification callback function is called.

event

The data transfer event that your application wishes to be notified of. See "Data Transfer Event Constants" (page 12) for a description of possible values. The type of event that can trigger your callback depends on the event mask you passed in the `eventRegister` parameter of the function `URLOpen` (page 46), and whether you pass a valid file specification in the `fileSpec` parameter of `URLOpen` (page 46). For more information, see the discussion.

callbackInfo

A pointer to a structure of type `URLCallbackInfo` (page 10). On return, the structure contains information about the data transfer event that occurred. The URL Access Manager passes this information to your callback function via the `callbackInfo` parameter of the function `InvokeURLNotifyUPP` (page 32).

Return Value

A result code. See "URL Access Manager Result Codes" (page 28). Your notification callback function should process the data transfer event and return `noErr`.

Discussion

Your notification callback function handles certain data transfer events that occur during data transfer operations performed by the function `URLOpen` (page 46). You can define an event notification function and the events for which you want to receive notification only if you do not specify a file in which to store the data for download operations. In order to be notified of these events, you must pass a UPP to your notification callback function in the `notifyProc` parameter. You indicate the type of data transfer events you want to receive via a bitmask in the `eventRegister` parameter.

Note that if you pass a valid file specification to URLOpen (page 46), your callback function will not be notified of data available and transaction completed events as identified by the constants kURLDataAvailableEvent and kURLTransactionCompleteEvent. If you pass a valid file specification to URLOpen (page 46), your callback function notified if any of the following events occur: kURLPercentEvent, kURLPeriodicEvent, kURLPropertyChangedEvent, kURLSystemEvent, kURLInitiatedEvent, kURLResourceFoundEvent, kURLDownloadingEvent, kURLUploadingEvent, kURLAbortInitiatedEvent, kURLCompletedEvent, and kURLErrorOccurredEvent.

When your callback is called, it should process the event immediately and return 0. You may wish your callback function to update its user interface, allocate and deallocate memory, or call the Thread Manager function NewThread.

Special Considerations

Do not call the function URLDisposeReference (page 36) from your notification callback function. Doing so may cause your application to stop working.

Availability

Available in Mac OS X v10.0 and later.

Declared In

URLAccess.h

URLSystemEventProcPtr

Defines a pointer to your system event notification callback that handles update events that occur while a dialog box is displayed during a data transfer operation.

```
typedef OSStatus (*URLSystemEventProcPtr)
(
    void * userContext,
    EventRecord * event
);
```

If you name your function MyURLSystemEventProc, you would declare it like this:

```
OSStatus MyURLSystemEventProc (
    void * userContext,
    EventRecord * event
);
```

Parameters

userContext

A pointer to application-defined storage that your application previously passed to the function URLSimpleDownload (page 49), URLDownload (page 37), URLSimpleUpload (page 51), or URLUpload (page 53). Your application can use this value to set up its context when the system event callback function is called.

event

A pointer to an event record containing information about the system event that occurred during the data transfer operation.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28). Your system event callback function should process the system event and return noErr.

Discussion
You pass a pointer to your callback function in the `eventProc` parameter of the function `URLSimpleDownload` (page 49), `URLSimpleUpload` (page 51), `URLDownload` (page 37), or `URLUpload` (page 53) if you want update events to be passed to your application while a dialog box is displayed by these functions. (In Mac OS X, this is not necessary, since all dialog boxes are moveable). In order for these functions to display a dialog box, you must set the mask constant `kURLDisplayProgressFlag` or `kURLDisplayAuthFlag` in the bitmask passed in the `openFlags` parameter. Call the function `NewURLSystemEventUPP` (page 35) to create a UPP to your system event notification callback. If you do not write your own system event notification callback, these functions will display a nonmovable modal dialog box.

When your callback is called, it should process the event immediately and return 0. You may wish your callback function to update its user interface, allocate and deallocate memory, or call the Thread Manager function `NewThread`.

Special Considerations

Do not call the function `URLDisposeReference` (page 36) from your callback function. Doing so may cause your application to stop working.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`URLAccess.h`

Data Types

URLCallbackInfo

Contains information about a data transfer event.

```
struct URLCallbackInfo {
    UInt32 version;
    URLReference urlRef;
    const char * property;
    UInt32 currentSize;
    EventRecord * systemEvent;
};
typedef struct URLCallbackInfo URLCallbackInfo;
```

Fields

`version`
The version of this structure. This value is currently 0.

`urlRef`
A reference to the URL associated with the data transfer event.

`property`

A pointer to a C string representing the name of the URL property that has changed, if relevant. This field is only valid if a property change event occurs as identified by the event constant `kURLPropertyChangedEvent`, described in "Data Transfer Event Constants" (page 12). or a description of name constants and data types of the corresponding property values, see "Universal URL Property Name Constants" (page 25) and "HTTP and HTTPS URL Property Name Constants" (page 24). You should specify this field if the event involves a change in a property value.

`currentSize`

The current total size (in bytes) of the data that has been downloaded and processed by the client.

`systemEvent`

A pointer to an event record containing information about the system event that occurred, if relevant. If the event is not a system event, as identified by the event constant `kURLSystemEvent`, described in "Data Transfer Event Constants" (page 12), this field is not valid.

Discussion
The `URLCallbackInfo` type represents a structure that contains information about the data transfer event that you want notification of. The URL Access Manager passes a pointer to this structure in the `callbackInfo` parameter of your notification callback function, see `URLNotifyProcPtr` (page 8).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`URLAccess.h`

URLNotifyUPP

```
typedef URLNotifyProcPtr URLNotifyUPP;
```

Discussion
For more information, see the description of the callback function `URLNotifyProcPtr` (page 8).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`URLAccess.h`

URLReference

Represents a reference to a URL.

```
typedef struct OpaqueURLReference * URLReference;
```

Discussion
The `URLReference` type represents a reference to an opaque structure that identifies a URL. You should call the function `URLNewReference` (page 45) to create a URL reference. The function `URLDisposeReference` (page 36) disposes of a URL reference when no longer needed. You pass a reference of this type to URL Access Manager functions that operate on a URL in some way.

Availability
Available in Mac OS X v10.0 and later.

Declared In
URLAccess.h

URLSystemEventUPP

```
typedef URLSystemEventProcPtr URLSystemEventUPP;
```

Discussion
For more information, see the description of the callback function URLSystemEventProcPtr (page 9).

Availability
Available in Mac OS X v10.0 and later.

Declared In
URLAccess.h

Constants

Authentication Type Constant

Represents the default value of the property value identified by the property name constant kURLAuthType.

```
enum {
    kUserNameAndPasswordFlag = 0x00000001
};
```

Constants
kUserNameAndPasswordFlag
Represents the default value of the property value identified by the property name constant kURLAuthType, described in "Universal URL Property Name Constants" (page 25). This value indicates that both the user name and password are used for authentication.

Available in Mac OS X v10.0 and later.

Declared in URLAccess.h.

Discussion
This constant represents the default value of the authentication type property value. The authentication type property value is identified by the property name constant kURLAuthType, described in "Universal URL Property Name Constants" (page 25). If you do not set the kURLAuthType property, the default value will be used for the authentication type. In this case, both the user name and password are used for authentication purposes.

Data Transfer Event Constants

Identify data transfer events that occur during a data transfer operation.

```
typedef UInt32 URLEvent;
enum {
    kURLInitiatedEvent = kURLInitiatingState,
    kURLResourceFoundEvent = kURLResourceFoundState,
    kURLDownloadingEvent = kURLDownloadingState,
    kURLAbortInitiatedEvent = kURLAbortingState,
    kURLCompletedEvent = kURLCompletedState,
    kURLErrorOccurredEvent = kURLErrorOccurredState,
    kURLDataAvailableEvent = kURLDataAvailableState,
    kURLTransactionCompleteEvent = kURLTransactionCompleteState,
    kURLUploadingEvent = kURLUploadingState,
    kURLSystemEvent = 29,
    kURLPercentEvent = 30,
    kURLPeriodicEvent = 31,
    kURLPropertyChangedEvent = 32
};
```

Constants

`kURLInitiatedEvent`

Indicates the function `URLOpen` (page 46) has been called but the location specified by the URL reference has not yet been accessed.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLResourceFoundEvent`

Indicates that the location specified by the URL reference has been accessed and is valid.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDownloadingEvent`

Indicates that a download operation is in progress.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLAbortInitiatedEvent`

Indicates that a data transfer operation has been aborted. When your application calls the function `URLAbort` (page 35), the URL Access Manager changes the state returned by the function `URLGetCurrentState` (page 39) to `kURLAbortingState` and passes the constant `kURLAbortInitiatedEvent` to your notification callback function. When data transfer is terminated, the URL Access Manager changes the state returned by `URLGetCurrentState` (page 39) to `kURLCompletedState` and passes the constant `kURLCompletedEvent` in the `event` parameter of your notification callback function.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLCompletedEvent`

Indicates that all operations associated with a call to `URLOpen` (page 46) have been completed. This includes the successful completion of a download or upload operation or the completion of cleanup work after aborting a download or upload operation. For example, when a data transfer operation is aborted, the URL Access Manager changes the state returned by the function `URLGetCurrentState` (page 39) to `kURLCompletedState` and passes the constant `kURLCompletedEvent` in the `event` parameter of your notification callback function.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLErrorOccurredEvent`
Indicates that an error occurred during data transfer. If you receive this event, you may wish to call the function `URLGetError` (page 41) to determine the nature of the error.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDataAvailableEvent`
Indicates that data is available in buffers. If you receive this event, you can call the function `URLGetBuffer` (page 38) to obtain the next buffer of data. You may wish to call the function `URLGetDataAvailable` (page 40) to determine the amount of data available for retrieval in a download operation. Note that if you pass a valid file specification in the `fileSpec` parameter of `URLOpen` (page 46), your notification callback function will not be called for data available events.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLTransactionCompleteEvent`
Indicates that a download operation is complete because there is no more data to retrieve from buffers. Note that if you pass a valid file specification in the `fileSpec` parameter of `URLOpen` (page 46), your notification callback function will not be called for transaction completed events.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLUploadingEvent`
Indicates that an upload operation is in progress.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLSystemEvent`
Indicates that a system event has occurred.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLPercentEvent`
Indicates that the size of the data being downloaded is known. In this case, an increment of one percent of the data was transferred into buffers.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLPeriodicEvent`
Indicates that a time interval of approximately one quarter of a second has passed.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLPropertyChangedEvent`
Indicates that a property such as a filename has become known or changed. In this case, the name of the changed property will be passed to your notification function via the `property` field of the `callbackInfo` structure.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

Discussion
The `URLEvent` enumeration defines constants that identify data transfer events that occur during a data transfer operation performed by `URLOpen` (page 46). In order to be notified of these events, you must pass a UPP to your data transfer notification callback in the `notifyProc` parameter. You indicate the type of data transfer events you want to receive via a bitmask in the `eventRegister` parameter. For a description of this bitmask, see "Data Transfer Event Mask Constants" (page 15).

Data Transfer Event Mask Constants

Represent a mask that identifies the data transfer events occurring during a data transfer operation that your application wants notification of.

```
typedef unsigned long URLEventMask;
enum {
    kURLInitiatedEventMask = 1 << (kURLInitiatedEvent - 1),
    kURLResourceFoundEventMask = 1 << (kURLResourceFoundEvent
- 1),
    kURLDownloadingMask = 1 << (kURLDownloadingEvent - 1),
    kURLUploadingMask = 1 << (kURLUploadingEvent - 1),
    kURLAbortInitiatedMask = 1 << (kURLAbortInitiatedEvent
- 1),
    kURLCompletedEventMask = 1 << (kURLCompletedEvent - 1),
    kURLErrorOccurredEventMask = 1 << (kURLErrorOccurredEvent
- 1),
    kURLDataAvailableEventMask = 1 << (kURLDataAvailableEvent
- 1),
    kURLTransactionCompleteEventMask = 1 << (kURLTransactionCompleteEvent
- 1),
    kURLSystemEventMask = 1 << (kURLSystemEvent - 1),
    kURLPercentEventMask = 1 << (kURLPercentEvent - 1),
    kURLPeriodicEventMask = 1 << (kURLPeriodicEvent - 1),
    kURLPropertyChangedEventMask = 1 << (kURLPropertyChangedEvent
- 1),
    kURLAllBufferEventsMask = kURLDataAvailableEventMask +
kURLTransactionCompleteEventMask,
    kURLAllNonBufferEventsMask = kURLInitiatedEventMask + kURLDownloadingMask
+ kURLUploadingMask + kURLAbortInitiatedMask + kURLCompletedEventMask
+ kURLErrorOccurredEventMask + kURLPercentEventMask + kURLPeriodicEventMask
+ kURLPropertyChangedEventMask,
    kURLAllEventsMask = 0xFFFFFFFF
};
```

Constants
`kURLInitiatedEventMask`
If the bit specified by this mask is set, your notification callback function will be notified when the function `URLOpen` (page 46) has been called but the location specified by the URL reference has not yet been accessed.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLResourceFoundEventMask`
If the bit specified by this mask is set, your notification callback function will be notified when the location specified by a URL reference has been accessed and is valid.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDownloadingMask`
If the bit specified by this mask is set, your notification callback function will be notified when a download operation is in progress.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLUploadingMask`
If the bit specified by this mask is set, your notification callback function will be notified when an upload operation is in progress.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLAbortInitiatedMask`
If the bit specified by this mask is set, your notification callback function will be notified when a download or upload operation has been aborted. When your application calls the function `URLAbort` (page 35), the URL Access Manager changes the state returned by the function `URLGetCurrentState` (page 39) to `kURLAbortingState` and passes the constant `kURLAbortInitiatedEvent` to your notification callback function. When data transfer is terminated, the URL Access Manager changes the state returned by `URLGetCurrentState` (page 39) to `kURLCompletedState` and passes the constant `kURLCompletedEvent` in the `event` parameter of your notification callback function.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLCompletedEventMask`
If the bit specified by this mask is set, your notification callback function will be notified when all operations associated with a call to the function `URLOpen` (page 46) have been completed. This indicates either the successful completion of an operation or the completion of cleanup work after aborting the operation. For example, when a data transfer operation is aborted, the URL Access Manager changes the state returned by the function `URLGetCurrentState` (page 39) to `kURLCompletedState` and passes the constant `kURLCompletedEvent` in the `event` parameter of your notification callback function.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLErrorOccurredEventMask`
If the bit specified by this mask is set, your notification callback function will be notified when an error has occurred. If you receive this event, you may wish to call the function `URLGetError` (page 41) to determine the nature of the error.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDataAvailableEventMask`

If the bit specified by this mask is set, your notification callback function will be notified when data is available in buffers. If you receive this event, you may wish to call the function `URLGetDataAvailable` (page 40) to determine the amount of data available for retrieval in a download operation. Note that if you pass a valid file specification in the `fileSpec` parameter of the function `URLOpen` (page 46), your notification callback function will not be called for data available events.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLTransactionCompleteEventMask`

If the bit specified by this mask is set, your notification callback function will be notified when the operation is complete because there is no more data to retrieve from buffers. Note that if you pass a valid file specification in the `fileSpec` parameter of the function `URLOpen` (page 46), your notification callback function will not be called for transaction completed events.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLSystemEventMask`

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLPercentEventMask`

If the bit specified by this mask is set, your notification callback function will be notified when an increment of one percent of the data has been transferred into buffers. This occurs only when the size of the data being transferred is known. This information is useful if you want the URL Access Manager to display a progress indicator.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLPeriodicEventMask`

If the bit specified by this mask is set, your notification callback function will be notified when a time interval of approximately one quarter of a second has passed. You can use this event to report the progress of the download operation when the size of the data is unknown or for other processing that you wish to perform at regular intervals.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLPropertyChangedEventMask`

If the bit specified by this mask is set, your notification callback function will be notified when the value of a URL property, such as a filename or user name, has become known or changes.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLAllBufferEventsMask`

If the bit specified by this mask is set, your notification callback function will be notified when a buffer-related event indicated by the event constants `kURLDataAvailableEvent` or `kURLTransactionCompleteEvent` occurred. If you pass a file specification in the `fileSpec` parameter of the function `URLOpen` (page 46), your notification callback function will not be called for buffer-related events.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLAllNonBufferEventsMask`

If the bit specified by this mask is set, your notification callback function will be notified when an event unrelated to a buffer occurred. This includes all events except those represented by the constants `kURLDataAvailableEvent` and `kURLTransactionCompleteEvent`.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLAllEventsMask`

If the bit specified by this mask is set, your notification callback function will be notified when any of the above data transfer events occur. If you pass a file specification in the `fileSpec` parameter of the function `URLOpen` (page 46), your notification callback function will not be called for buffer-related events.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

Discussion

The `URLEventMask` enumeration defines masks that identify the data transfer events occurring during a call to the function `URLOpen` (page 46) that your application wants notification of. For a description of data transfer events, see "Data Transfer Event Constants" (page 12). You can define an event notification function and the events for which you want to receive notification only if you do not specify a file in which to store the data for downloads. You can indicate which events you want to receive notification of via a bitmask in the `eventRegister` parameter of `URLOpen` (page 46).

Data Transfer Options Mask Constants

Represent a mask that identifies the data transfer options to use when uploading or downloading data.

```
typedef UInt32 URLOpenFlags;
enum {
    kURLReplaceExistingFlag = 1 << 0,
    kURLBinHexFileFlag = 1 << 1,
    kURLExpandFileFlag = 1 << 2,
    kURLDisplayProgressFlag = 1 << 3,
    kURLDisplayAuthFlag = 1 << 4,
    kURLUploadFlag = 1 << 5,
    kURLIsDirectoryHintFlag = 1 << 6,
    kURLDoNotTryAnonymousFlag = 1 << 7,
    kURLDirectoryListingFlag = 1 << 8,
    kURLExpandAndVerifyFlag = 1 << 9,
    kURLNoAutoRedirectFlag = 1 << 10,
    kURLDebinhexOnlyFlag = 1 << 11,
    kURLDoNotDeleteOnErrorFlag = 1 << 12,
    kURLResumeDownloadFlag = 1 << 13,
    kURLReservedFlag = (unsigned long) 1 << 31
};
```

Constants

`kURLReplaceExistingFlag`

If the bit specified by this mask is set and the destination file or directory exists, the file or directory contents are replaced by the newly downloaded or uploaded data. If this bit is not set and the name of the file is specified and does exist, the URL Access Manager returns the result code `kURLDestinationExistsError`. If the name of the file or directory is not specified, the file or directory already exists, and the bit specified by this mask is not set, a number is appended to the name before any extension until a unique name is created, and the data is transferred to the new file or directory name without notifying the calling application that the name has changed. In the case of a download operation, your application can check the `destination` parameter of the functions `URLSimpleDownload` (page 49) and `URLDownload` (page 37) to obtain the new filename.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLBinHexFileFlag`

If the bit specified by this mask is set, the URL Access Manager converts a nontext file that has a resource fork to `BinHex` format before uploading it.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLExpandFileFlag`

If the bit specified by this mask is set, files in `BinHex` format are decoded. If version 5.0 of the Stuffit Engine is installed in the System Folder, the URL Access Manager uses it to expand the file.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDisplayProgressFlag`

If the bit specified by this mask is set, the URL Access Manager displays a nonmovable modal progress indicator during data transfer operations only if you have not provided a system event notification callback. On Mac OS X, dialog boxes will always be moveable. To handle data transfer events that occur while a progress indicator is being displayed, pass a UPP to your data transfer event notification callback in the `eventProc` parameter of the functions `URLSimpleDownload` (page 49), `URLDownload` (page 37), `URLSimpleUpload` (page 51), and `URLUpload` (page 53).

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDisplayAuthFlag`

If the bit specified by this mask is set, the URL Access Manager displays a nonmovable modal authentication dialog box when user authentication is required only if you have not provided a system event notification callback. On Mac OS X, dialog boxes will always be moveable. To handle data transfer events that occur while an authentication dialog box is being displayed, pass a UPP to your data transfer event notification callback in the `eventProc` parameter of the functions `URLSimpleDownload` (page 49), `URLDownload` (page 37), `URLSimpleUpload` (page 51), and `URLUpload` (page 53). If the bit specified by this mask is clear, the user name and password properties of the URL are used for authentication purposes. If these are not set, the URL Access Manager returns the result code `kURLAuthenticationError`.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLUploadFlag`

If the bit specified by this mask is set, the function `URLOpen` (page 46) will upload the file or directory to the specified URL.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLIsDirectoryHintFlag`

If the bit specified by this mask is set, download operations assume that the URL points to a directory. Note that if you pass a pathname that specifies a file in the `url` parameter of the function `URLSimpleDownload` (page 49), the file is downloaded regardless of whether you specify `kURLDirectoryListingFlag` or `KURLIsDirectoryHintFlag` in the `openFlags` parameter.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDoNotTryAnonymousFlag`

If the bits specified by this mask is set, when FTP authentication occurs, the functions `URLSimpleDownload` (page 49), `URLDownload` (page 37), `URLSimpleUpload` (page 51), `URLUpload` (page 53), and `URLOpen` (page 46) will not try to log on anonymously. Instead, they will rely on the setting of the mask constant `kURLDisplayAuthFlag`. If the bit specified by the `kURLDoNotTryAnonymousFlag` mask is not set, these functions will first attempt to log on anonymously.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDirectoryListingFlag`

If the bit specified by this mask is set, a listing of the directory, rather than the entire directory, is downloaded. If the URL points to a file instead of a directory, the file is downloaded. Note that if you pass a pathname that specifies a file in the `url` parameter of the function `URLSimpleDownload` (page 49), the file is downloaded regardless of whether you specify `kURLDirectoryListingFlag` or `KURLIsDirectoryHintFlag` in the `openFlags` parameter.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLExpandAndVerifyFlag`

If this flag is available (that is, the File Signing shared library is available) and the bit specified by this mask is set, the signature attached to the file is verified. Success indicates that the file was signed by the certificate authority, but the certificate will not be displayed until after the file is downloaded.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLNoAutoRedirectFlag`

If the bit specified by this mask is set, if an HTTP request returns a "redirect" status (300, 301, or 302), the transfer will complete without attempting to redirect to the next URL. Otherwise, redirects are followed until actual data is encountered or an error is returned.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDebinhexOnlyFlag`

If the bit specified by this mask is set, the internal engine is used to decode files, rather than the external Stuffit Engine, even if Stuffit is installed. This prevents the display of the Stuffit progress user interface. If you set this bit, you must also set the `kURLExpandFileFlag` mask.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDoNotDeleteOnErrorFlag`

Do not delete the downloaded file if an error or abort occurs. This flag applies to downloading only and should be used if interested in later resuming the download.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLResumeDownloadFlag`

The passed in file is partially downloaded, attempt to resume it. Currently works for HTTP only. If no `FSSpec` passed in, this flag will be ignored. Overridden by `kURLReplaceExistingFlag`.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLReservedFlag`

Reserved for internal use.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

Discussion

The `URLOpenFlags` enumeration defines masks you can use to identify the data transfer options you want used when performing data transfer operations. You pass this mask in the `openFlags` parameter of the functions `URLSimpleDownload` (page 49), `URLDownload` (page 37), `URLSimpleUpload` (page 51), `URLUpload` (page 53), and `URLOpen` (page 46). The options that you can specify for upload and download operations differ, as do those that you can specify for the low-level function `URLOpen` (page 46). For a description of the options you can specify in each case, see the appropriate function discussions.

Data Transfer State Constants

Identifies the current state of a data transfer operation.

```
typedef UInt32 URLState;
enum {
    kURLNullState = 0,
    kURLInitiatingState = 1,
    kURLLookingUpHostState = 2,
    kURLConnectingState = 3,
    kURLResourceFoundState = 4,
    kURLDownloadingState = 5,
    kURLDataAvailableState = 0x10 + kURLDownloadingState,
    kURLTransactionCompleteState = 6,
    kURLErrorOccurredState = 7,
    kURLAbortingState = 8,
    kURLCompletedState = 9,
    kURLUploadingState = 10
};
```

Constants

`kURLNullState`

Indicates that the function `URLOpen` (page 46) has not yet been called.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLInitiatingState`

Indicates that the function `URLOpen` (page 46) has been called, but the location specified by the URL reference has not yet been accessed. The stream enters this state from the `kURLNullState` state.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLLookingUpHostState`

Indicates that the function `URLOpen` (page 46) has been called, and that the host is being looked up. The stream enters this state from the `kURLInitiatingState` state.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLConnectingState`

Indicates that the function `URLOpen` (page 46) has been called, and a connection is being made. The stream enters this state from the `kURLLookingUpHostState` state.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLResourceFoundState`

Indicates that the location specified by the URL reference has been accessed and is valid. The stream enters this state from the `kURLConnectingState` state.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDownloadingState`

Indicates that the download operation is in progress but there is currently no data in the buffers. The stream enters this state initially from the `kURLResourceFoundState` state. During a download operation, the stream's state may alternate between the `kURLDownloadingState` and the `kURLDataAvailableState` states.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLDataAvailableState`

Indicates that the download operation is in progress and data is available in the buffers. The stream initially enters this state from the `kURLDownloadingState` state. During a download operation, the stream's state may alternate between the `kURLDownloadingState` and the `kURLDataAvailableState` states. If the stream is in the data available state, you may want to call the function `URLGetDataAvailable` (page 40) to determine the amount of data available for download. If you pass `NULL` in the `fileSpec` parameter of the function `URLOpen` (page 46), you will need to call the function `URLGetBuffer` (page 38) to obtain the next buffer of data.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLTransactionCompleteState`

Indicates that a download or upload operation is complete. The stream can enter this state from the `kURLDownloadingState` state.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLErrorOccurredState`

Indicates that an error occurred during data transfer. The stream can enter this state from any state except the `kURLAbortingState` state. If the stream is in this state, you may wish to call the function `URLGetError` (page 41) to determine the nature of the error.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLAbortingState`

Indicates that a download or upload operation is aborting. The stream enters this state from the `kURLErrorOccurredState` state or as a result of calling the function `URLOpen` (page 46) when the stream is in any other state. When your application calls the function `URLAbort` (page 35), the URL Access Manager changes the state returned by the function `URLGetCurrentState` (page 39) to `kURLAbortingState` and passes the constant `kURLAbortInitiatedEvent` to your notification callback function.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLCompletedState`

Indicates that there is no more activity to be performed on this stream. In this case, the data transfer has either completed successfully or been aborted. The stream enters this state from the `kURLTransactionCompleteState` or the `kURLAbortingState` state. When data transfer is terminated after a data transfer operation is aborted, the URL Access Manager changes the state returned by `URLGetCurrentState` (page 39) to `kURLCompletedState` and passes the constant `kURLCompletedEvent` in the `event` parameter of your notification callback function.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLUploadingState`

Indicates that an upload operation is in progress.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

Discussion

The `URLState` enumeration defines constants that identify the status of a data transfer operation with respect to a URL. The function `URLGetCurrentState` (page 39) passes back one of these constants in the `state` parameter to indicate the status of a data transfer operation. All constants except `kURLDataAvailableState`

and `kURLCompletedState` can be returned at any time. If you pass a valid file specification in the `fileSpec` parameter of the function `URLOpen` (page 46), your notification callback function will not be notified of data available and transaction completed states as identified by the constants `kURLDataAvailableState` and `kURLTransactionCompleteState`.

HTTP and HTTPS URL Property Name Constants

Identify property values specific to HTTP and HTTPS URLs.

```
#define kURLHTTPRequestMethod "URLHTTPRequestMethod"
#define kURLHTTPRequestHeader "URLHTTPRequestHeader"
#define kURLHTTPRequestBody "URLHTTPRequestBody"
#define kURLHTTPRespHeader "URLHTTPRespHeader"
#define kURLHTTPUserAgent "URLHTTPUserAgent"
#define kURLHTTPRedirectedURL "URLHTTPRedirectedURL"
#define kURLSSLCipherSuite "URLSSLCipherSuite"
```

Constants

`kURLHTTPRequestMethod`

Identifies the HTTP request method property value. You use this name constant to set or obtain a C string that represents the HTTP method to be used in the request. If you are posting a form, you must set this property to the string "POST".

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLHTTPRequestHeader`

Identifies the HTTP request header property value. You use this name constant to set or obtain a C string that represents the HTTP header to be used in the request. You may set this property to contain all headers needed for the request. If you are posting a form and have set the properties identified by the name constants `kURLHTTPRequestMethod` and `kURLHTTPRequestBody`, you do not need to set the property identified by this tag.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLHTTPRequestBody`

Identifies the HTTP request body property value. You use this name constant to set or obtain a buffer of data that represents the HTTP body to be provided in the request. If you set the property identified by this tag but not that identified by the name constant `kURLHTTPHeader`, a body-length header is automatically added to the request. If you are posting a form, you must set this property to the form data you want sent.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLHTTPRespHeader`

Identifies the HTTP response header property value. You use this name constant to obtain a C string that represents the HTTP response header that was received.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLHTTPUserAgent`

Identifies the user agent property value. You use this name constant to set or obtain a C string that represents the HTTP user agent string that is embedded in HTTP requests. By default, the URL Access Manager sets the user agent string to "URL Access 1.0 (Macintosh; PPC)".

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLHTTPRedirectedURL`

Identifies the redirected URL property value. You use this name constant to obtain a C string that represents the URL that you were redirected to.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLSSLCipherSuite`

Identifies the SSL cipher suite property value.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

Discussion

These constants represent Apple-defined name constants that identify property values specific to HTTP and HTTPS URLs. For a description of the name constants that identify property values universal to all URLs, see `Universal URL Property Name Constants` (page 25).

You pass one of these name constants in the property parameter of the functions `URLSetProperty` (page 48) and `URLGetProperty` (page 42), respectively, to set or obtain a particular property value. Note that you can only set HTTP and HTTPS property values identified by the constants `kURLHTTPRequestMethod`, `kURLHTTPRequestHeader`, `kURLHTTPRequestBody`, and `kURLHTTPUserAgent`. You must also pass the correct data type corresponding to the property value in the `propertyBuffer` parameter of these functions.

Version Notes

Prior to version 2.0.3 of the URL Access Manager, the data type of the property value identified by the name constant `kURLHTTPRequestBody` was a C string. In 2.0.3 and later, the data type is a buffer of data.

Universal URL Property Name Constants

Identify property values universal to all URLs.

```
#define kURLURL "URLString"
#define kURLResourceSize "URLResourceSize"
#define kURLLastModifiedTime "URLLastModifiedTime"
#define kURLMIMEType "URLMIMEType"
#define kURLFileType "URLFileType"
#define kURLFileCreator "URLFileCreator"
#define kURLCharacterSet "URLCharacterSet"
#define kURLResourceName "URLResourceName"
#define kURLHost "URLHost"
#define kURLAuthType "URLAuthType"
#define kURLUserName "URLUserName"
#define kURLPassword "URLPassword"
#define kURLStatusString "URLStatusString"
#define kURLIsSecure "URLIsSecure"
#define kURLCertificate "URLCertificate"
#define kURLTotalItems "URLTotalItems"
#define kURLConnectTimeout "URLConnectTimeout"
```

Constants

`kURLURL`

Identifies the name string property value. You use this name constant to obtain a C string that represents the URL.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLResourceSize`

Identifies the resource size property value. You use this name constant to obtain a value of type `Size` that represents the total size of the data at the location specified by the URL.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLLastModifiedTime`

Identifies the modification time property value. You use this name constant to obtain a value of type `UInt32` that represents the last time the data was modified.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLMIMEType`

Identifies the MIME type property value. You use this name constant to obtain a Pascal string that represents the MIME type of the URL.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLFileType`

Identifies the file type property value. You use this name constant to set or obtain a value of type `OSType` that represents the file type as specified in a call to the function `URLOpen` (page 46). If the file type was not specified, `kURLFileType` obtains the file type compatible with the MIME type.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLFileCreator`
Identifies the file creator property value. You use this name constant to set or obtain a value of type `OSType` that represents the file creator as specified in a call to the function `URLOpen` (page 46). If the file creator was not specified, `kURLFileType` obtains the file type compatible with the MIME type.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLCharacterSet`
Identifies the character set property value. You use this name constant to obtain a Pascal string that represents the character set used by the URL, as returned by the HTTP server.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLResourceName`
Identifies the resource name property value. You use this name constant to obtain a Pascal string that represents the name associated with the data to be downloaded.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLHost`
Identifies the host property value. You use this name constant to obtain a Pascal string that represents the host on which the data is located.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLAuthType`
Identifies the authentication type property value. You use this name constant to obtain a value that represents the type of authentication that the download operation requires. The default authentication type is `kUserNameAndPasswordFlag`, described in `Authentication Type Constant` (page 12).

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLUserName`
Identifies the user name property value. You use this name constant to set or obtain a Pascal string that represents the user name used for authentication.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLPassword`
Identifies the password property value. You use this name constant to set or obtain a Pascal string that represents the password used for authentication.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLStatusString`
Identifies the status property value. You use this name constant to obtain a Pascal string that represents the current status of the data stream. You can use this property to display the status of the data transfer operation.

Available in Mac OS X v10.0 and later.

Declared in `URLAccess.h`.

`kURLIsSecure`
: Identifies the security property value. You use this name constant to get a Boolean value that indicates whether the download operation is secure.

  Available in Mac OS X v10.0 and later.

  Declared in `URLAccess.h`.

`kURLCertificate`
: Identifies the certificate property value. You use this name constant to obtain a buffer of data that represents the certificate provided by a remote server.

  Available in Mac OS X v10.0 and later.

  Declared in `URLAccess.h`.

`kURLTotalItems`
: Identifies the total items property value. You use this name constant to obtain a value of type `UInt32` that represents the total number of items being uploaded or downloaded.

  Available in Mac OS X v10.0 and later.

  Declared in `URLAccess.h`.

`kURLConnectTimeout`
: Identifies the connection timeout property value.

  Available in Mac OS X v10.0 and later.

  Declared in `URLAccess.h`.

Discussion
These constants represent Apple-defined name constants that identify property values universal to all URLs. For a description of the name constants that identify property values specific to HTTP and HTTPS URLs, see `HTTP and HTTPS URL Property Name Constants` (page 24).

You pass one of these name constants in the property parameter of the functions `URLSetProperty` (page 48) and `URLGetProperty` (page 42), respectively, to set or obtain a particular property value. Note that you can only set the universal property values identified by the constants `kURLPassword` and `kURLUserName`. You must also pass the correct data type corresponding to the property value in the `propertyBuffer` parameter of these functions.

Result Codes

The most common result codes returned by URL Access Manager are listed in the table below. The following result codes may also be returned; `noErr` (0), `nsvErr` (-35), `fnfErr` (-43), `paramErr` (-50), and `dirNFErr` (-120).

Result Code	Value	Description
`kURLInvalidURLReferenceError`	-30770	Returned by functions that operate on URL references to indicate that a reference is invalid. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kURLProgressAlreadyDisplayedError	-30771	Returned by the functions URLSimpleDownload, URLDownload, URLSimpleUpload, and URLUpload to indicate that a progress indicator is already displayed. Available in Mac OS X v10.0 and later.
kURLDestinationExistsError	-30772	Returned by the functions URLSimpleUpload and URLUpload to indicate that the destination file already exists. Available in Mac OS X v10.0 and later.
kURLInvalidURLError	-30773	Returned by functions that operate on URL strings to indicate that the format of the URL is invalid. Available in Mac OS X v10.0 and later.
kURLUnsupportedSchemeError	-30774	Returned by functions that operate on URL strings to indicate that the transfer protocol is not supported. Available in Mac OS X v10.0 and later.
kURLServerBusyError	-30775	Indicates a failed data transfer operation. Available in Mac OS X v10.0 and later.
kURLAuthenticationError	-30776	Returned by URLSimpleDownload, URLDownload, URLSimpleUpload, and URLUpload functions if no authentication dialog box is allowed and the user name and password properties of the URL are set incorrectly or don't exist. Available in Mac OS X v10.0 and later.
kURLPropertyNotYetKnownError	-30777	Returned by the functions URLGetProperty and URLGetPropertySize to indicate that the value or size of a URL property is not available. Available in Mac OS X v10.0 and later.
kURLUnknownPropertyError	-30778	Returned by functions URLSetProperty, URLGetProperty, and URLGetPropertySize to indicate that the property is invalid or undefined. Available in Mac OS X v10.0 and later.
kURLPropertyBufferTooSmallError	-30779	Returned by the function URLGetProperty to indicate that the buffer is too small to receive the requested property. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kURLUnsettablePropertyError	-30780	Returned by the function URLSetProperty to indicate that the property cannot be set. Available in Mac OS X v10.0 and later.
kURLInvalidCallError	-30781	Returned by the functions URLGetDataAvailable, URLGetBuffer, and URLReleaseBuffer to indicate that the call is invalid. Available in Mac OS X v10.0 and later.
kURLFileEmptyError	-30783	Indicates a failed data transfer operation. Available in Mac OS X v10.0 and later.
kURLExtensionFailureError	-30785	Indicates that your extension failed to load. Available in Mac OS 9 and earlier.
kURLInvalidConfigurationError	-30786	Indicates a failed data transfer operation. This is returned when you attempt to upload through an HTTP proxy, since upload through proxies is not supported. Available in Mac OS X v10.0 and later.
kURLAccessNotAvailableError	-30787	Returned by the function URLGetURLAccessVersion to indicate that the URL Access Manager is not available. Available in Mac OS X v10.0 and later.
kURL68kNotSupportedError	-30788	Indicates that URL Access Manager was called from within a 68K context. Available in Mac OS 9 and earlier.

Deprecated URL Access Manager Reference (Not Recommended) Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.4

DisposeURLNotifyUPP

Disposes of a UPP to your data transfer event notification callback. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
void DisposeURLNotifyUPP (
   URLNotifyUPP userUPP
);
```

Parameters

userUPP

A Universal Procedure Pointer (UPP) to your notification callback function.

Discussion

When you are finished with a UPP to your notification callback function, you should dispose of it by calling the `DisposeURLNotifyUPP` function.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

DisposeURLSystemEventUPP

Disposes of a UPP to your system event notification callback. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
void DisposeURLSystemEventUPP (
   URLSystemEventUPP userUPP
);
```

Parameters

userUPP

A UPP to your system event callback function.

Discussion

When you are finished with a UPP to your system event callback function, you should dispose of it by calling the `DisposeURLSystemEventUPP` function.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

InvokeURLNotifyUPP

Invokes your data transfer event notification callback. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus InvokeURLNotifyUPP (
   void *userContext,
   URLEvent event,
   URLCallbackInfo *callbackInfo,
   URLNotifyUPP userUPP
);
```

Parameters

userContext

A pointer to application-defined storage. The URL Access Manager passes this value in the `userContext` parameter of your notification callback function. Your application can use this to set up its context when your data transfer event notification callback is called.

event

The data transfer events you want your application to receive. See "Data Transfer Event Constants" (page 12) for a description of possible values. The URL Access Manager tests the bitmask you pass in the `eventRegister` parameter of the function `URLOpen` (page 46) to determine which events to pass to your callback function. See "Data Transfer Event Mask Constants" (page 15) for a description of this bitmask.

callbackInfo

A pointer to a structure of type `URLCallbackInfo` (page 10) that provides information about the data transfer event to your callback function. The URL Access Manager passes a pointer to this structure in the `callbackInfo` parameter of your notification callback function.

userUPP

A Universal Procedure Pointer to your data transfer notification callback. For information on how to write this function, see `URLNotifyProcPtr` (page 8).

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The URL Access Manager calls the `InvokeURLNotifyUPP` function when you pass a UPP to your callback function in the `notifyProc` parameter of the function `URLOpen` (page 46), and the data transfer event that you specified in the `eventRegister` parameter occurs.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

InvokeURLSystemEventUPP

Invokes your system event notification callback. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus InvokeURLSystemEventUPP (
   void *userContext,
   EventRecord *event,
   URLSystemEventUPP userUPP
);
```

Parameters

userContext

A pointer to application-defined storage. The URL Access Manager passes this value in the `userContext` parameter of your system event callback function. Your application can use this to set up its context when your system event notification callback is called.

event

A pointer to an event record that provides information about the system event to your callback function.

userUPP

A Universal Procedure Pointer to your system event notification callback. For information on how to write this function, see `URLSystemEventProcPtr` (page 9).

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The URL Access Manager calls the `InvokeURLSystemEventUPP` function when you pass a UPP to your callback function in the eventProc parameter of the functions `URLSimpleDownload` (page 49), `URLSimpleUpload` (page 51), `URLDownload` (page 37), or `URLUpload` (page 53), and a system event occurs while a progress indicator or authentication dialog box is being displayed.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

NewURLNotifyUPP

Creates a UPP to your data transfer event notification callback. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
URLNotifyUPP NewURLNotifyUPP (
   URLNotifyProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your data transfer event notification callback. For information on how to write your callback, see `URLNotifyProcPtr` (page 8).

Return Value

A UPP to your data transfer event notification callback. You can register your callback by passing this UPP in the `notifyProc` parameter of the function `URLOpen` (page 46). See the description of the `URLNotifyUPP` data type.

Discussion

The `NewURLNotifyUPP` function creates a pointer to your data transfer event notification callback. You pass a pointer to your callback in the `notifyProc` parameter of the function `URLOpen` (page 46) if you want your application to receive data transfer events. Pass a bitmask in the `eventRegister` parameter of `URLOpen` (page 46) indicating which data transfer events you want to receive.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

When you are finished with your data transfer event notification callback, you should dispose of the UPP by calling the function `DisposeURLNotifyUPP` (page 31).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

NewURLSystemEventUPP

Creates a UPP to your system event notification callback. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
URLSystemEventUPP NewURLSystemEventUPP (
   URLSystemEventProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your system event notification callback. For information on how to write your callback, see `URLSystemEventProcPtr` (page 9).

Return Value

A UPP to your system event notification callback. You can register your callback by passing this UPP in the `eventProc` parameter of the function `URLSimpleDownload` (page 49), `URLSimpleUpload` (page 51), `URLDownload` (page 37), or `URLUpload` (page 53). See the description of the `URLSystemEventUPP` data type.

Discussion

The `NewURLSystemEventUPP` function creates a pointer to your system event callback function. You pass a pointer to your callback function in the `eventProc` parameter of the functions `URLSimpleDownload` (page 49), `URLSimpleUpload` (page 51), `URLDownload` (page 37), and `URLUpload` (page 53) if you want update events to be passed to your application while a dialog box is displayed. (In Mac OS X, this is not necessary, since all dialog boxes are moveable). In order for these functions to display a dialog box, you must set the mask constant `kURLDisplayProgressFlag` or `kURLDisplayAuthFlag` in the bitmask passed in the `openFlags` parameter.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

When you are finished with your system event notification callback, you should dispose of the UPP by calling the function `DisposeURLNotifyUPP` (page 31).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

URLAbort

Terminates a data transfer operation. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLAbort (
   URLReference urlRef
);
```

Parameters

urlRef

A reference to the URL whose data transfer operation you wish to terminate.

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The `URLAbort` function terminates any data transfer operation started by the functions `URLSimpleDownload` (page 49), `URLDownload` (page 37), `URLSimpleUpload` (page 51), `URLUpload` (page 53), or `URLOpen` (page 46). When your application calls `URLAbort`, the URL Access Manager changes the state returned by the function `URLGetCurrentState` (page 39) to `kURLAbortingState` and passes the constant `kURLAbortInitiatedEvent` to your notification callback function. When data transfer is terminated, the URL Access Manager changes the state returned by `URLGetCurrentState` (page 39) to `kURLCompletedState` and passes the constant `kURLCompletedEvent` in the `event` parameter of your notification callback function.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLDisposeReference

Disposes of the memory associated with a URL reference. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLDisposeReference (
   URLReference urlRef
);
```

Parameters

urlRef

A reference to the URL whose associated memory you wish to dispose of. You should call the `URLDisposeReference` function to release the memory occupied by a URL reference when you are finished with it.

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

You must call the `URLDisposeReference` function to dispose of the reference associated with a URL reference even if the data transfer operation fails. Failure to call `URLDisposeReference` may result in thread or memory leaks.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
URLAccess.h

URLDownload

Downloads data from a URL specified by a URL reference. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLDownload (
   URLReference urlRef,
   FSSpec *destination,
   Handle destinationHandle,
   URLOpenFlags openFlags,
   URLSystemEventUPP eventProc,
   void *userContext
);
```

Parameters

urlRef

A reference to the URL from which data is to be downloaded. Once you call URLDownload, you cannot use the same reference if you call URLDownload again. Instead, you must create a new URL reference by calling the function URLNewReference (page 45).

destination

A pointer to a file specification structure that identifies the file or directory into which data is to be downloaded. If you wish to download data into memory, pass NULL in this parameter and a valid handle in the destinationHandle parameter. If you pass a file specification that does not identify a file or directory, the name of the file or directory specified by the pathname in the urlRef parameter is used. If you pass a file or directory that already exists, and do not specify kURLReplaceExistingFlag in the openFlags parameter, URLDownload creates a new file or directory whose name has a number appended before the extension. For example, if the URL specifies a file named file.txt, URLDownload changes the filename to file1.txt.

destinationHandle

A handle to the destination in memory where you want the data downloaded. Before calling URLDownload, create a zero-sized handle. If you wish to download data into a file or directory, pass NULL in this parameter and a valid file specification in the destination parameter.

openFlags

A bitmask that indicates the data transfer options to use. You can specify any of the following masks for downloading options: kURLReplaceExistingFlag, kURLExpandFileFlag, kURLExpandAndVerifyFlag, kURLDisplayProgressFlag, kURLDisplayAuthFlag, kURLIsDirectoryHintFlag, kURLDoNotTryAnonymousFlag, kURLDebinhexOnlyFlag, kURLNoAutoRedirect, and kURLDirectoryListingFlag. See "Data Transfer Options Mask Constants" (page 18) for a description of possible values.

eventProc

A Universal Procedure Pointer (UPP) to your system event callback function, if one exists. For information on how to write a system event callback, see URLSystemEventProcPtr (page 9). If you want to handle events that occur while a progress indicator or authentication dialog box is being displayed, specify the appropriate mask (either kURLDisplayProgressFlag or kURLDisplayAuthFlag) in the openFlags parameter and pass a UPP to your callback function in this parameter. Pass NULL if you do not want to receive notification of these events. In this case, the URL Access Manager displays a nonmovable modal progress indicator or authentication dialog box.

userContext

A pointer to application-defined storage that will be passed to your system event callback function, if one exists. Your application can use this to set up its context when your system event callback function is called.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28). If your application is multi-threaded, and more than one thread calls `URLDownload` simultaneously, `URLDownload` returns the result code `kURLProgessAlreadyDisplayedError` if you specify `kURLDisplayProgressFlag` in the `openFlags` parameter and the URL Access Manager is already displaying a progress indicator.

Discussion

The `URLDownload` function downloads data from a URL specified by a URL reference to a file, directory, or memory. It does not return until the download is complete. If you want to download data from a URL identified by a URL string rather than a reference, call the function `URLSimpleDownload` (page 49). The difference between the two functions is that URLDownload allows you to access other URL Access Manager functions before, after, or during the download. If you want more control over a data transfer operation, call the function `URLOpen` (page 46).

If you wish to download data to a file or directory, pass a valid file specification in the `destination` parameter. If you instead wish to download data to memory, pass a valid handle in the `destinationHandle` parameter. If the URL specified in the `urlRef` parameter points to a file, the file is downloaded regardless of whether the bit specified by the mask constant `kURLDirectoryListingFlag` or `KURLIsDirectoryHintFlag` is set in the `openFlags` parameter.

When `URLDownload` downloads data from a URL that represents a local file (that is, a URL that begins with `file://`), the data fork is downloaded but the resource fork is not.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

`URLDownload` yields time to other threads. Your application should call `URLDownload` from a thread other than the main thread so that other processes have time to run.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

URLGetBuffer

Obtains the next buffer of data in a download operation. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLGetBuffer (
   URLReference urlRef,
   void **buffer,
   Size *bufferSize
);
```

Parameters

urlRef

A reference to the URL whose next buffer you wish to obtain.

buffer

On return, a handle to a buffer containing the downloaded data.

bufferSize

On return, a pointer to the number of bytes of data in the buffer.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28).

Discussion

The `URLGetBuffer` function obtains the next buffer of data in a download operation. `URLGetBuffer` does not enable you to retain or modify the transferred data. If you pass `NULL` in the `fileSpec` parameter of the function `URLOpen` (page 46), you should call `URLGetBuffer` to retrieve data as it is downloaded.

You should call `URLGetBuffer` repeatedly until URL Access Manager passes the event constant `kURLCompletedEvent` or `kURLAbortInitiatedEvent` in the `event` parameter of your notification callback function, or until the function `URLGetCurrentState` (page 39) returns the state constant `kURLTransactionComplete` or `kURLAbortingState`. Between calls to `URLGetBuffer`, you should call the function `URLIdle` (page 45) to allow time for the URL Access Manager to refill its buffers.

To determine the number of bytes remaining in the buffer, call the function `URLGetDataAvailable` (page 40). The size returned by `URLGetDataAvailable` (page 40) does not include the number of bytes in transit to a buffer, nor does it include the amount of data not yet transferred from the URL.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

You should release the returned buffer as soon as possible after a call to `URLGetBuffer` by calling the function `URLReleaseBuffer` (page 48). This prevents the URL Access Manager from running out of buffers.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

URLGetCurrentState

Determines the status of a data transfer operation. (Deprecated in Mac OS X v10.4.)

```
OSStatus URLGetCurrentState (
   URLReference urlRef,
   URLState *state
);
```

Parameters

urlRef

A reference to the URL whose data transfer state you want to determine.

state

On return, a pointer to the state of data transfer. See "Data Transfer State Constants" (page 21) for a description of possible values.

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The `URLGetCurrentState` function determines the current status of a data transfer operation. You may wish to call `URLGetCurrentState` periodically to monitor the status of a download or upload operation.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLGetDataAvailable

Determines the amount of data currently available for retrieval in a download operation. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLGetDataAvailable (
   URLReference urlRef,
   Size *dataSize
);
```

Parameters

urlRef

A reference to the URL for which you wish to determine the amount od data currently available for retrieval in a download operation.

dataSize

On return, a pointer to the size (in bytes) of data available for retrieval in a download operation.

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The `URLGetDataAvailable` (page 40) function determines the amount of data remaining in the buffer of the URL Access Manager that you will obtain from a call to the function `URLGetBuffer` (page 38). You should only call this function if you passed an invalid destination file to the function `URLOpen` (page 46).

This does not include the number of bytes in transit to your buffer, nor does it include the amount of data not yet transferred from the URL Access Manager. To calculate the amount of data remaining to be downloaded, pass the name constant `kURLResourceSize` in the `property` parameter of the function `URLGetProperty` (page 42) and subtract the amount of data copied.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLGetError

Determines the error code of a failed data transfer operation. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLGetError (
   URLReference urlRef,
   OSStatus *urlError
);
```

Parameters

urlRef

A reference to the URL whose data transfer operation failed.

urlError

A pointer to a C string representing the name of the error code returned by the failed operation.

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The `URLGetError` function determines the error code returned when a data transfer operation fails. The error code may be a system error code, a protocol-specific error code, or one of the error codes listed in "URL Access Manager Result Codes" (page 28).

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLGetFileInfo

Obtains the file type and creator of a file. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLGetFileInfo (
   StringPtr fName,
   OSType *fType,
   OSType *fCreator
);
```

Parameters

fName

A pointer to a Pascal string representing the name of the file for which you want information.

fType

On return, a pointer to the file type code of the specified filename.

fCreator

On return, a pointer to the file creator code of the specified filename.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28).

Discussion

The `URLGetFileInfo` function obtains the file type and creator codes for a specified filename. The type and creator codes are determined by the Internet configuration mapping table and are based on the filename extension. For example, if you pass the filename `"jane.txt"`, `URLGetFileInfo` will return `'TEXT'` in the `type` parameter and `'ttxt'` in the `creator` parameter.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

URLGetProperty

Obtains the value of a URL property. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLGetProperty (
   URLReference urlRef,
   const char *property,
   void *propertyBuffer,
   Size bufferSize
);
```

Parameters

urlRef

A reference to the URL whose property value you want to determine.

property

A pointer to a C string representing the name of the property value you want to determine. For a description of property name constants and their corresponding data types, see "Universal URL Property Name Constants" (page 25) and "HTTP and HTTPS URL Property Name Constants" (page 24).

propertyBuffer

A pointer to a buffer containing the property value you want to obtain. You must also pass the correct data type of the property value you wish to obtain. Before calling `URLGetProperty`, allocate enough memory in this buffer to contain the property value you wish to obtain. On return, a pointer to a buffer containing the property value. If you do not allocate enough memory for the buffer, `URLGetProperty` does not pass back the property value in this parameter and returns the result code `kURLPropertyBufferTooSmallError`.

bufferSize

The size (in bytes) of the buffer pointed to by `propertyBuffer`. To determine the buffer size, call the function `URLGetPropertySize` (page 43). If the buffer size is too small, `URLGetProperty` returns the result code `kURLPropertyBufferTooSmallError` and does not pass back the property value in the `propertyBuffer` parameter.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28). The result code `kURLPropertyBufferTooSmallError` indicates that you did not allocate enough memory for the buffer in the `propertyBuffer` parameter. The result code `kURLPropertyNotYetKnownError` indicates that the value of the property is not yet available.

Discussion

The `URLGetProperty` function obtains the value of a URL property identified by the property name constant specified in the property parameter.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

URLGetPropertySize

Determines the size of a URL property. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLGetPropertySize (
   URLReference urlRef,
   const char *property,
   Size *propertySize
);
```

Parameters

urlRef

A reference to the URL whose property size you want to determine.

property

A pointer to a C string representing the name of the property value whose size you want to determine. For a description of property name constants, see "Universal URL Property Name Constants" (page 25) and "HTTP and HTTPS URL Property Name Constants" (page 24).

propertySize

On return, a pointer to the size (in bytes) of the specified property value. If the size is not available, `URLGetPropertySize` passes back `-1` in this parameter and returns the result code `kURLPropertyNotYetKnownError`.

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The `URLGetProperty` function obtains the size of the property value identified by the property name constant passed in the property parameter. For a description of property name constants and data types of the corresponding property values, see "Universal URL Property Name Constants" (page 25) and "HTTP and HTTPS URL Property Name Constants" (page 24).

You should call the `URLGetPropertySize` function before calling the function `URLGetProperty` (page 42) to determine the size of the buffer containing the property value you wish to obtain. Pass the value passed back in the `propertySize` parameter in the `bufferSize` parameter of `URLGetProperty` (page 42).

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLGetURLAccessVersion

Determines the version of URL Access Manager installed on the user's system. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLGetURLAccessVersion (
   UInt32 *returnVers
);
```

Parameters

returnVers

On return, a pointer to the version number of the URL Access Manager installed on the user's system.

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLIdle

Gives the URL Access Manager time to refill its buffers during download operations. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLIdle (
   void
);
```

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The `URLIdle` function gives the URL Access Manager time to refill its buffers during download operations. You should call URLIdle periodically after you call the function `URLOpen` (page 46) to allow time for the URL Access Manager to refill its buffers.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLNewReference

Creates a URL reference. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLNewReference (
   const char *url,
   URLReference *urlRef
);
```

Parameters

url
A pointer to a C string representing the name of the URL you want to create.

urlRef
On return, a pointer to the newly-created URL reference.

Return Value
A result code. See "URL Access Manager Result Codes" (page 28).

Discussion
The `URLNewReference` function creates a URL reference that you can use in subsequent calls to the URL Access Manager. When you no longer need a URL reference, you should dispose of its memory by calling the function `URLDisposeReference` (page 36).

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLOpen

Opens a URL and starts an asynchronous download or upload operation. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLOpen (
   URLReference urlRef,
   FSSpec *fileSpec,
   URLOpenFlags openFlags,
   URLNotifyUPP notifyProc,
   URLEventMask eventRegister,
   void *userContext
);
```

Parameters

urlRef

A reference to the URL to or from which you wish to transfer data. You cannot use the same reference if you call `URLOpen` again. Instead, you must create a new URL reference by calling the function `URLNewReference` (page 45). If the URL refers to a file, the file is downloaded regardless of whether you specify `kURLDirectoryListingFlag` or `KURLIsDirectoryHintFlag` in the `openFlags` parameter. See "Naming Your Destination File" for more information.

fileSpec

A pointer to a file specification that identifies the file or directory from which data is to be uploaded or downloaded. For upload operations, you must pass a valid file specification. For download operations, you can pass `NULL`. In this case, you must call the function `URLGetBuffer` (page 38) to retrieve the data as it is downloaded. For more information, see the function discussion.

openFlags

A bitmask that indicates the data transfer options to use. You can specify any of the following masks for upload operations: `kURLUploadFlag`, `kURLReplaceExistingFlag`, `kURLBinHexFileFlag`, and `kURLDoNotTryAnonymousFlag`. You can specify any of the following masks for download operations: `kURLReplaceExistingFlag`, `kURLIsDirectoryHintFlag`, `kURLDoNotTryAnonymousFlag`, `kURLDebinhexOnlyFlag`, `kURLNoAutoRedirect`, and `kURLDirectoryListingFlag`. See "Data Transfer Options Mask Constants" (page 18) for a description of possible values.

notifyProc

A Universal Procedure Pointer (UPP) to a data transfer event notification callback, as described in `URLNotifyProcPtr` (page 8). You should create a notification callback function if you wish to receive notification of certain data transfer events. In this case, you should also pass a bitmask of the events you wish to receive in the `eventRegister` parameter. The data transfer events that you receive will vary depending upon whether the destination file you specify is valid. Pass `NULL` if you do not want to receive notification of data transfer events.

eventRegister

A bitmask that `URLOpen` will test to determine the data transfer events that you wish to receive notification of. To receive data transfer events, you should also pass a UPP to your callback in the `notifyProc` parameter. See "Data Transfer Event Mask Constants" (page 15) for a description of this mask.

userContext

A pointer to application-defined storage that will be passed to your notification callback function. Your application can use this to set up its context when your notification callback function is called.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28).

Discussion

The `URLOpen` function starts a download or upload operation and returns control to your application immediately. For download operations, you do not have to specify a valid destination file. In this case, you should call the function `URLGetBuffer` (page 38) repeatedly to get the next buffer of data. Between calls to `URLGetBuffer` (page 38), you should call the function `URLIdle` (page 45) to allow time for the URL Access Manager to refill its buffers during download operations. After each call to `URLGetBuffer` (page 38), you call the function `URLReleaseBuffer` (page 48) to prevent the URL Access Manager from running out of buffers. You can call the function `URLGetDataAvailable` (page 40) to determine the amount of data remaining in the buffer of the URL Access Manager that you will obtain from a call to the function `URLGetBuffer` (page 38).

If you pass a valid destination file, you should not call the functions `URLGetBuffer` (page 38), `URLReleaseBuffer` (page 48), or `URLGetDataAvailable` (page 40).

If you wish to be notified of certain data transfer events, you can specify a data transfer event callback and pass a pointer to it in the `URLEventMask` parameter of URLOpen. The data transfer events that you receive will vary depending upon whether the destination file you specify is valid. In addition, you should pass a bitmask representing the events you wish to be notified of in the `eventRegister` parameter.

When `URLOpen` downloads data from a URL that represents a local file (that is, a URL that begins with `file://`), the data fork is downloaded but the resource fork is not.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

URLReleaseBuffer

Releases a buffer. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLReleaseBuffer (
   URLReference urlRef,
   void *buffer
);
```

Parameters

urlRef

A reference to the URL whose buffer you want to release.

buffer

A pointer to the buffer you want to release.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28).

Discussion

The `URLReleaseBuffer` function releases the buffer obtained by calling the function `URLGetBuffer` (page 38). To prevent the URL Access Manager from running out of buffers, you should call `URLReleaseBuffer` after each call to `URLGetBuffer` (page 38).

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

URLSetProperty

Sets the value of a URL property. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLSetProperty (
   URLReference urlRef,
   const char *property,
   void *propertyBuffer,
   Size bufferSize
);
```

Parameters

urlRef

A reference to the URL whose property value you want to set.

property

A pointer to a C string representing the name of the property value you want to set. You can only set property values identified by the constants `kURLPassword`, `kURLUserName`, `kURLHTTPRequestMethod`, `kURLHTTPRequestHeader`, `kURLHTTPRequestBody`, and `kURLHTTPUserAgent`. For a description of these property name constants and their corresponding data types, see "Universal URL Property Name Constants" (page 25) and "HTTP and HTTPS URL Property Name Constants" (page 24).

propertyBuffer

A pointer to a buffer containing the data you would like the property to be set to. The data must be of the correct type.

bufferSize

The size (in bytes) of the data you want to set.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28). The result code `kURLUnsettablePropertyError` indicates that a property value cannot be set.

Discussion

The `URLSetProperty` function enables you to set those property values identified by the following constants: `kURLPassword`, `kURLUserName`, `kURLPassword`, `kURLHTTPRequestMethod`, `kURLHTTPRequestHeader`, `kURLHTTPRequestBody`, and `kURLHTTPUserAgent`. For a description of these property name constants and their corresponding data types, see "Universal URL Property Name Constants" (page 25) and "HTTP and HTTPS URL Property Name Constants" (page 24).

You may wish to call `URLSetProperty` before calling the function `URLDownload` (page 37) or `URLUpload` (page 53) to set a URL property before a data transfer operation.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

URLSimpleDownload

Downloads data from a URL specified by a character string. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLSimpleDownload (
   const char *url,
   FSSpec *destination,
   Handle destinationHandle,
   URLOpenFlags openFlags,
   URLSystemEventUPP eventProc,
   void *userContext
);
```

Parameters

url

A pointer to a C string representing the pathname of the URL from which data is to be downloaded. If the pathname specifies a file, the file is downloaded regardless of whether you specify `kURLDirectoryListingFlag` or `KURLIsDirectoryHintFlag` in the `openFlags` parameter.

destination

A pointer to a file specification structure that identifies the file or directory into which data is to be downloaded. If you wish to download data into memory, pass `NULL` in this parameter and a valid handle in the `destinationHandle` parameter. If you pass a file specification that does not identify a file or directory, the name of the file or directory specified by the pathname in the `url` parameter is used. If you pass a file or directory that already exists, and do not specify `kURLReplaceExistingFlag` in the `openFlags` parameter, `URLSimpleDownload` creates a new file or directory whose name has a number appended before the extension. For example, if the URL specifies a file named `file.txt`, `URLSimpleDownload` changes the filename to `file1.txt`.

destinationHandle

A handle to the destination in memory where you want the data downloaded. Before calling `URLDownload`, create a zero-sized handle. If you wish to download data into a file or directory, pass `NULL` in this parameter and a valid file specification in the `destination` parameter.

openFlags

A bitmask that indicates the data transfer options to use. You can specify any of the following masks for downloading options: `kURLReplaceExistingFlag`, `kURLExpandFileFlag`, `kURLExpandAndVerifyFlag`, `kURLDisplayProgressFlag`, `kURLDisplayAuthFlag`, `kURLIsDirectoryHintFlag`, `kURLDoNotTryAnonymousFlag`, `kURLDebinhexOnlyFlag`, `kURLNoAutoRedirect`, and `kURLDirectoryListingFlag`. See "Data Transfer Options Mask Constants" (page 18) for a description of possible values.

eventProc

A Universal Procedure Pointer (UPP) to your system event callback function, if one exists. For information on how to write a system event callback, see `URLSystemEventProcPtr` (page 9). If you want to handle events that occur while a progress indicator or authentication dialog box is being displayed, specify the appropriate mask (either `kURLDisplayProgressFlag` or `kURLDisplayAuthFlag`) in the `openFlags` parameter and pass a UPP to your callback function in this parameter. Pass `NULL` if you do not want to receive notification of these events. In this case, the URL Access Manager displays a nonmovable modal progress indicator or authentication dialog box.

userContext

A pointer to application-defined storage that will be passed to your system event callback function, if one exists. Your application can use this to set up its context when your system event callback function is called.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28). If your application is multi-threaded, and more than one thread calls `URLSimpleDownload` simultaneously, `URLSimpleDownload` returns the result code `kURLProgressAlreadyDisplayedError` if you specify `kURLDisplayProgressFlag` in the `openFlags` parameter and the URL Access Manager is already displaying a progress indicator.

Discussion
The `URLSimpleDownload` function downloads data from a URL specified by a pathname to a specified file, directory, or memory. It does not return until the download is complete. If you want to download data from a URL identified by a reference rather than a pathname, call the function `URLDownload` (page 37). The difference between the two functions is that `URLDownload` (page 37) allows you to access other URL Access Manager functions before, after, or during the download. If you want more control over a data transfer operation, call the function `URLOpen` (page 46).

If you wish to download data to a file or directory, pass a valid file specification in the `destination` parameter. If you instead wish to download data to memory, pass a valid handle in the `destinationHandle` parameter.

When `URLSimpleDownload` downloads data from a URL that represents a local file (that is, a URL that begins with `file://`), the data fork is downloaded but the resource fork is not.

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

`URLSimpleDownload` yields time to other threads. Your application should call URLSimpleDownload from a thread other than the main thread so that other processes have time to run.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLSimpleUpload

Uploads a file or directory to an FTP URL specified by a character string. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLSimpleUpload (
   const char *url,
   const FSSpec *source,
   URLOpenFlags openFlags,
   URLSystemEventUPP eventProc,
   void *userContext
);
```

Parameters

url

A pointer to a C string representing the URL to which a file or directory is to be uploaded. If you wish to replace the destination directory of this URL with the file or directory that you pass in the source parameter, terminate the string with a slash character (/), and set the mask constant `kURLReplaceExistingFlag` in the `openFlags` parameter. If you specify a name that already exists on the server and do not specify `kURLReplaceExistingFlag`, `URLSimpleUpload` returns the result code `kURLDestinationExistsError`. If you do not specify a name, do not specify `kURLReplaceExistingFlag` in the `openFlags` parameter, and the name already exists on the server, the URL Access Manager creates a unique name by appending a number to the original name before the extension, if any. For example, if the URL specifies a file named `file.txt`, `URLSimpleUpload` changes the filename to `file1.txt`. See "Naming Your Destination File" for more information.

source

A pointer to a file specification structure that describes the file or directory you want to upload.

openFlags

A bitmask that indicates the data transfer options to use. You can specify any of the following masks for uploading options: `kURLReplaceExistingFlag`, `kURLBinHexFileFlag`, `kURLDisplayProgressFlag`, `kURLDisplayAuthFlag`, and `kURLDoNotTryAnonymousFlag`. See "Data Transfer Options Mask Constants" (page 18) for a description of possible values.

eventProc

A Universal Procedure Pointer (UPP) to your system event callback function, if one exists. For information on how to write a system event callback, see `URLSystemEventProcPtr` (page 9). If you want to handle events that occur while a progress indicator or authentication dialog box is being displayed, specify the appropriate mask (either `kURLDisplayProgressFlag` or `kURLDisplayAuthFlag`) in the `openFlags` parameter and pass a UPP to your callback function in this parameter. Pass `NULL` if you do not want to receive notification of these events. In this case, the URL Access Manager displays a nonmovable modal progress indicator or authentication dialog box.

userContext

A pointer to application-defined storage that will be passed to your system event callback function, if one exists. Your application can use this to set up its context when your system event callback function is called.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28). The result code `kURLDestinationExistsError` indicates that you specified a pathname that already exists on the server but did not set the bit specified by the mask constant `kURLReplaceExistingFlag` in the `openFlags` parameter. If your application is multi-threaded, and more than one thread calls `URLSimpleUpload` simultaneously, `URLSimpleUpload` returns the result code `kURLProgessAlreadyDisplayedError` if you specify `kURLDisplayProgressFlag` in the `openFlags` parameter and the URL Access Manager is already displaying a progress indicator.

Discussion
The `URLSimpleUpload` function uploads a file or directory to an FTP URL specified by a pathname. It does not return until the upload is complete. If you want to upload data from a URL identified by a reference rather than a pathname, call the function `URLUpload` (page 53). The difference between the two functions is that `URLUpload` (page 53) allows you to access other URL Access Manager functions before, after, or during the download. If you want more control over a data transfer operation, call the function `URLOpen` (page 46).

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

`URLSimpleUpload` yields time to other threads. Your application should call `URLSimpleUpload` from a thread other than the main thread so that other processes have time to run.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`URLAccess.h`

URLUpload

Uploads a file or directory to an FTP URL specified by a URL reference. (Deprecated in Mac OS X v10.4. Use CFNetwork instead; see *CFNetwork Programming Guide*.)

```
OSStatus URLUpload (
   URLReference urlRef,
   const FSSpec *source,
   URLOpenFlags openFlags,
   URLSystemEventUPP eventProc,
   void *userContext
);
```

Parameters

urlRef

A reference to the URL to which a file or directory is to be uploaded. Once you have called `URLUpload`, you cannot use the same reference again. If you wish to replace the destination directory of this URL with the file or directory that you pass in the source parameter, set the mask constant `kURLReplaceExistingFlag` in the `openFlags` parameter. If you specify a name that already exists on the server and do not specify `kURLReplaceExistingFlag`, `URLUpload` returns the result code `kURLDestinationExistsError`. If you do not specify a name, do not specify `kURLReplaceExistingFlag` in the `openFlags` parameter, and the name already exists on the server, the URL Access Manager creates a unique name by appending a number to the original name before the extension, if any. For example, if the URL specifies a file named `file.txt`, `URLUpload` changes the filename to `file1.txt`. See "Naming Your Destination File" for more information.

source

A pointer to a file specification structure that describes the file or directory you want to upload.

openFlags

A bitmask that indicates the data transfer options you want used. You can specify any of the following masks for uploading options: `kURLReplaceExistingFlag`, `kURLBinHexFileFlag`, `kURLDisplayProgressFlag`, `kURLDisplayAuthFlag`, and `kURLDoNotTryAnonymousFlag`. See "URL Access Manager Reference" (page ?) for a description of possible values.

eventProc

A Universal Procedure Pointer (UPP) to your system event callback function, if one exists. For information on how to write a system event callback, see `URLSystemEventProcPtr` (page 9). If you want to handle events that occur while a progress indicator or authentication dialog box is being displayed, specify the appropriate mask (either `kURLDisplayProgressFlag` or `kURLDisplayAuthFlag`) in the `openFlags` parameter and pass a UPP to your callback function in this parameter. Pass `NULL` if you do not want to receive notification of these events. In this case, the URL Access Manager displays a nonmovable modal progress indicator or authentication dialog box.

userContext

A pointer to application-defined storage that will be passed to your system event callback function, if one exists. Your application can use this to set up its context when your system event callback function is called.

Return Value

A result code. See "URL Access Manager Result Codes" (page 28). The result code `kURLDestinationExistsError` indicates that you specified a pathname that already exists on the server but did not specify `kURLReplaceExistingFlag` in the `openFlags` parameter. If your application is multi-threaded, and more than one thread calls `URLUpload` simultaneously, `URLUpload` returns the result code `kURLProgessAlreadyDisplayedError` if you specify `kURLDisplayProgressFlag` in the `openFlags` parameter and the URL Access Manager is already displaying a progress indicator.

Discussion

The `URLUpload` function uploads a file or directory to an FTP URL specified by a URL reference. It does not return until the upload is complete. If you want to upload data from a URL identified by a pathname rather than a reference, call the function `URLSimpleUpload` (page 51). The difference between the two functions is that URLUpload allows you to access other URL Access Manager functions before, after, or during the download. If you want more control over a data transfer operation, call the function `URLOpen` (page 46).

Special Considerations

CFNetwork provides better reliability and performance and is used by Apple's own applications. URL Access Manager is no longer being enhanced or improved.

`URLUpload` yields time to other threads. Your application should call `URLUpload` from a thread other than the main thread so that other processes have time to run.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`URLAccess.h`

Document Revision History

This table describes the changes to *URL Access Manager Reference*.

Date	Notes
2006-07-13	Updated for Mac OS X v10.5.
2006-07-24	Added deprecation information to individual functions.
2005-07-07	URLAccess Manager is deprecated as of Mac OS X v10.4. Use CFNetwork instead.
2002-12-01	Updated the EDD.
2001-07-01	Last version of this document.

Document Revision History

Index

N

U